

9. Design of Compact Linguistic Models

As in the case of pattern classification problems, genetic algorithm-based rule selection and genetics-based machine learning can be applied to the design of linguistic rule-based systems for modeling problems [85, 90]. These two schemes for pattern classification problems are slightly modified in this chapter to apply them to modeling problems. As in the previous chapter, we use linguistic rules of the following form to approximately realize an n -input and single-output nonlinear function:

$$\text{Rule } R_q: \text{ If } x_1 \text{ is } A_{q1} \text{ and } \dots \text{ and } x_n \text{ is } A_{qn} \text{ then } y \text{ is } B_q. \quad (9.1)$$

Our task in this chapter is to design a linguistic rule-based system from the given m input-output pairs (\mathbf{x}_p, y_p) , $p = 1, 2, \dots, m$, where $\mathbf{x}_p = (x_{p1}, x_{p2}, \dots, x_{pn})$ is an n -dimensional input vector and y_p is the corresponding output value. For simplicity of explanation, we assume that the input-output space has already been normalized into the unit hypercube $[0, 1]^{n+1}$. We also assume that K linguistic terms are given for each of the n input and single output variables.

9.1 Single-Objective and Multi-Objective Formulations

9.1.1 Three Objectives in the Design of Linguistic Models

In Chap. 6, we explained the three-objective optimization problem in the design of linguistic rule-based systems for pattern classification problems. The three objectives were the classification accuracy, the number of linguistic rules, and the total rule length of linguistic rules. The first objective should be modified for modeling problems while the other two objectives can be used with no modifications.

Let S be a set of linguistic rules of the form (9.1). In addition to the given K linguistic terms, we use *don't care* as an antecedent fuzzy set (i.e., A_{qi} in (9.1)). This special fuzzy set is not used as a consequent fuzzy set (i.e., B_q in (9.1)) because linguistic rules with *don't care* in the consequent part are meaningless. Thus the total number of possible linguistic rules is $K(K+1)^n$. The rule set S is a subset of these linguistic rules. The rule set S can be viewed as a fuzzy rule-based system for our modeling problem.

We measure the accuracy of the rule set S by the total squared error as

$$f_1(S) = \sum_{p=1}^m \{\hat{y}(\mathbf{x}_p) - y_p\}^2 / 2, \quad (9.2)$$

where $\hat{y}(\mathbf{x}_p)$ is the estimated output value for the input vector $\mathbf{x}_p = (x_{p1}, x_{p2}, \dots, x_{pn})$ by the rule set S . We can use any fuzzy reasoning method for the calculation of $\hat{y}(\mathbf{x})$. In this chapter, we use the non-standard fuzzy reasoning method in (8.50) of the previous chapter. Note that the non-standard fuzzy reasoning method is the same as the standard fuzzy reasoning method in (8.32) when no inclusion relation holds among linguistic rules in the rule set S .

Since S is an arbitrary subset of the $K(K+1)^n$ linguistic rules, there are many cases where the entire input space is not covered by the rule set S . This means that the estimated output value $\hat{y}(\mathbf{x})$ is not always calculated for an arbitrary input vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$. When there is no compatible linguistic rule in S for the input vector \mathbf{x}_p , the corresponding estimated output value $\hat{y}(\mathbf{x}_p)$ cannot be calculated from S . In this case, the squared error for the input-output pair (\mathbf{x}_p, y_p) cannot be calculated in (9.2), either. Thus we use a pre-specified penalty value as the squared error when $\hat{y}(\mathbf{x}_p)$ cannot be calculated:

$$(\hat{y}(\mathbf{x}_p) - y_p)^2 = \delta^2, \quad (9.3)$$

where δ is a pre-specified positive constant. In our computer simulations in this chapter, we specified the penalty value as $(\hat{y}(\mathbf{x}_p) - y_p)^2 = 1$ because the output value y_p is normalized into a real number in the unit interval $[0, 1]$.

As in Chap. 6 for pattern classification problems, the second objective $f_2(S)$ and the third objective $f_3(S)$ are the number of linguistic rules in S (i.e., $|S|$) and the total rule length of linguistic rules in S , respectively.

Using the three objectives, the design of linguistic rule-based systems for modeling problems is formulated as

$$\text{Minimize } f_1(S), \text{ minimize } f_2(S), \text{ and minimize } f_3(S). \quad (9.4)$$

Note that all three objectives are to be minimized. In Chap. 6, the first objective was to be maximized because it was the number of correctly classified training patterns by the rule set S . In this chapter, $f_1(S)$ should be minimized because it is the total squared error.

9.1.2 Handling as a Single-Objective Optimization Problem

When the weight for each objective is available from a human user, the three objectives in (9.4) can be combined into a single scalar objective function as

$$\text{Minimize } f(S) = w_1 \cdot f_1(S) + w_2 \cdot f_2(S) + w_3 \cdot f_3(S), \quad (9.5)$$

where w_1 , w_2 , and w_3 are non-negative real numbers. The three weights w_1 , w_2 , and w_3 should be specified according to the user's preference with

respect to the three objectives. We assume that the weight values are given by the human user. The minimization problem in (9.5) can be treated in the framework of single-objective optimization. Thus standard optimization techniques are applicable to the design of linguistic rule-based systems. This is an advantage of the single-objective formulation in (9.5) over the multi-objective formulation in (9.4).

The main drawback of the single-objective formulation is related to the specification of the weight values with respect to the three objectives. It is not easy for the human user to appropriately specify the weight values according to their preference with respect to the three objectives. Moreover, the final solution (i.e., the obtained rule set) strongly depends on the specification of the weight values. This dependency is illustrated in Fig. 9.1. For simplicity of explanation, the three-dimensional objective space is represented as a two-dimensional objective space in Fig. 9.1 where the ellipsoidal region shows all subsets (i.e., all rule sets) of the linguistic rules. Figure 9.1 shows the relation between the search direction and the obtained rule set. The search direction is specified by the three weights w_1 , w_2 , and w_3 . When the weight w_1 with respect to the total squared error is much larger than the other two weights w_2 and w_3 , a complicated rule set with high accuracy will be obtained (e.g., the rule set S_a will be obtained from the search direction \mathbf{w}_a in Fig. 9.1). In this case, the obtained rule set may consist of a large number of long linguistic rules. On the other hand, when the two weights w_2 and w_3 with respect to the complexity of rule sets are much larger than the other weight w_1 for the total squared error, a simple rule set with low accuracy may be obtained (e.g., the rule set S_b will be obtained from the search direction \mathbf{w}_b). In this case, the obtained rule set may consist of a small number of short linguistic rules. When the three weights are of the same magnitude, a compromise solution may be obtained (e.g., the rule set S_c will be obtained from the search direction \mathbf{w}_c). From these discussions, we can see that the obtained rule set from the single-objective formulation strongly depends on the specification of the three weight values. In real-world applications, the single-objective optimization problem in (9.5) will be solved several times using different weight vectors to find a number of alternative rule sets.

9.1.3 Handling as a Three-Objective Optimization Problem

As in Chap. 6, we can use multi-objective optimization algorithms to find non-dominated rule sets with respect to the three objectives in (9.4). Let us briefly review the concept of non-dominated rule sets for the three-objective optimization problem in (9.4). A rule set S_B is said to dominate another rule set S_A (i.e., S_B is better than S_A) if all the following inequalities hold:

$$f_1(S_A) \geq f_1(S_B), \quad f_2(S_A) \geq f_2(S_B), \quad f_3(S_A) \geq f_3(S_B), \quad (9.6)$$

and at least one of the following inequalities holds:

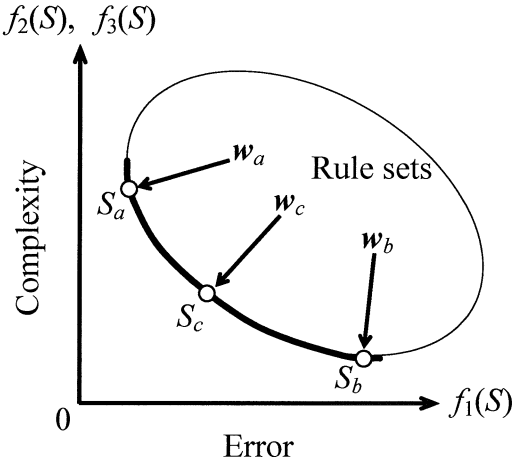


Fig. 9.1. Relation between the search direction and the obtained rule set in the multi-dimensional objective space. The bold curve shows the Pareto front of the multi-objective optimization problem

$$f_1(S_A) > f_1(S_B), \quad f_2(S_A) > f_2(S_B), \quad f_3(S_A) > f_3(S_B). \quad (9.7)$$

The first condition (i.e., all three inequalities in (9.6)) means that no objective of S_B is worse than S_A (i.e., S_B is not worse than S_A). The second condition (i.e., one of the three inequalities in (9.7)) means that at least one objective of S_B is better than S_A . When a rule set S is not dominated by any other rule set, S is said to be a Pareto-optimal solution of the three-objective optimization problem in (9.4). In Fig. 9.1, Pareto-optimal solutions are rule sets on the bold curve. The above conditions in (9.6)–(9.7) are slightly different from Chap. 6. This is because the first objective is to be minimized in this chapter while it was to be maximized in Chap. 6.

When the search space is not large, it may be easy to find all the Pareto-optimal solutions of the three-objective optimization problem in (9.4). On the other hand, when the search space is huge, it is impractical to try to find true Pareto-optimal solutions. In this case, multi-objective optimization algorithms try to find near-optimal solutions. Non-dominated solutions among examined ones are presented to the human user as search results.

In this chapter, we use the multi-objective genetic algorithm (MOGA) in Sect. 6.2 after modifying the definition of the fitness function. For the three-objective optimization problem in (9.4), the fitness value of each rule set S is defined as

$$\text{fitness}(S) = -w_1 \cdot f_1(S) - w_2 \cdot f_2(S) - w_3 \cdot f_3(S), \quad (9.8)$$

where w_1 , w_2 , and w_3 are weights satisfying the following conditions:

$$w_1, w_2, w_3 \geq 0, \quad (9.9)$$

$$w_1 + w_2 + w_3 = 1. \quad (9.10)$$

Since all three objectives are to be minimized, a negative sign is added to each weight in the fitness function in (9.8). The fitness function is supposed

to be maximized. The other parts of the MOGA in Sect. 6.2 are used for modeling problems with no modifications.

9.2 Multi-Objective Rule Selection

As we have already explained in Chaps. 4 and 6 for pattern classification problems, genetic algorithm-based rule selection consists of two phases: candidate rule generation and rule selection. In the first phase, a large number of candidate rules are generated. When too many candidate rules are generated, prescreening is used to decrease the number of candidate rules. In the second phase, a small number of linguistic rules are selected from a large number of candidate rules to design a linguistic rule-based system. In the framework of single-objective optimization, a single rule set is obtained. On the other hand, multiple rule sets are obtained as non-dominated solutions of the three-objective optimization problem in (9.4) when we used the MOGA. In this section, we explain a genetic algorithm-based rule selection method that is designed to find multiple non-dominated rule sets of the three-objective optimization problem in (9.4).

9.2.1 Candidate Rule Generation

As we have explained, the total number of combinations of antecedent and consequent fuzzy sets is $K(K+1)^n$ when we use K linguistic terms and *don't care* for each of the n input variables and K linguistic terms for the single output variable. Thus the total number of linguistic rules is also $K(K+1)^n$. For low-dimensional modeling problems, we can use all the $K(K+1)^n$ linguistic rules as candidate rules in rule selection.

Let us consider a two-input and single-output nonlinear function in Fig. 9.2. This nonlinear function was depicted using nine linguistic rules in Fig. 9.3. When the three linguistic terms (i.e., S: *small*, M: *medium*, and L: *large*) are given for the two input and single output variables as in Fig. 9.3, the total number of combinations of antecedent and consequent fuzzy sets is $3 \times (3+1)^2 = 48$. Thus the total number of possible linguistic rules is also 48. Genetic algorithms can easily handle such a small number of linguistic rules as candidate rules.

9.2.2 Candidate Rule Prescreening

Candidate rule prescreening is a procedure for decreasing the number of candidate rules in a heuristic manner. As we have explained in Chap. 4, candidate rule prescreening significantly improves the efficiency of genetic algorithm-based rule selection. A simple prescreening procedure for modeling problems is to remove linguistic rules with no compatible training data. That is, this

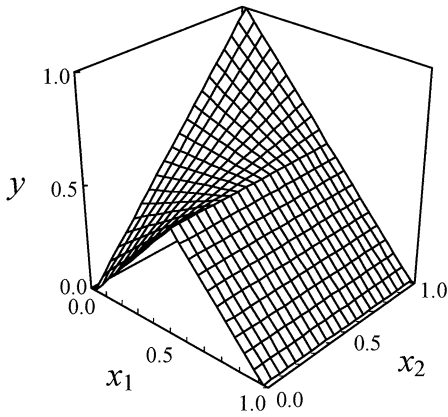


Fig. 9.2. A nonlinear function used as a numerical example. This nonlinear function was depicted from the nine linguistic rules in Fig. 9.3

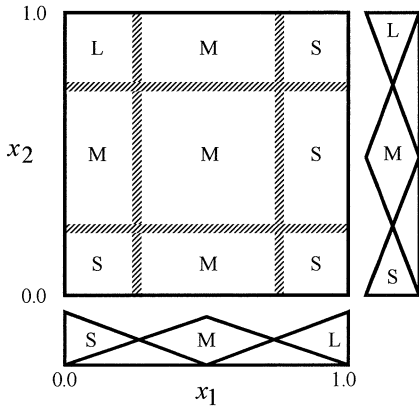


Fig. 9.3. Nine linguistic rules used for depicting the nonlinear function in Fig. 9.2

prescreening procedure removes linguistic rules that satisfy the following relation:

$$\mu_{R_q}(\mathbf{x}_p, y_p) = 0, \quad p = 1, 2, \dots, m. \quad (9.11)$$

The compatibility grade of each input–output pair (\mathbf{x}_p, y_p) with the linguistic rule R_q is defined by its antecedent part A_q and the consequent part B_q as

$$\mu_{R_q}(\mathbf{x}_p, y_p) = \mu_{A_q}(\mathbf{x}_p) \times \mu_{B_q}(y_p). \quad (9.12)$$

Let us assume that 441 input–output pairs are generated as training data from the nonlinear function in Fig. 9.2 using the uniformly divided 21×21 grid of the two-dimensional input space $[0, 1] \times [0, 1]$. While we can generate the 48 candidate rules by considering all the possible combinations of antecedent and consequent fuzzy sets, some rules have no compatible training data. For example, we can see from Fig. 9.2 that the following linguistic rule has no compatible training data:

$$\text{If } x_1 \text{ is } \textit{large} \text{ and } x_2 \text{ is } \textit{large} \text{ then } y \text{ is } \textit{large}. \quad (9.13)$$

Among the possible 48 candidate rules, 8 rules have no compatible training data. Thus the removal of these rules is likely to have no bad effect on the performance of the rule sets finally obtained by genetic algorithm-based rule selection.

For high-dimensional problems, we cannot examine all the $K(K+1)^n$ combinations of antecedent and consequent fuzzy sets for generating candidate rules. This is because the number of these combinations (i.e., $K(K+1)^n$) increases exponentially as the number of input variables (i.e., n) increases. As in Chap. 4, we can use the rule length as a heuristic prescreening criterion for decreasing the number of examined linguistic rules. The total number of possible linguistic rules of length L for an n -input and single-output nonlinear function is $K \cdot {}_nC_L \cdot K^L$ where K is the number of consequent linguistic terms, ${}_nC_L$ is the number of combinations of choosing L out of n input variables, and K^L is the number of combinations of K antecedent linguistic terms for L input variables. The number of short linguistic rules is not large even when the total number of linguistic rules is huge (i.e., $K \cdot {}_nC_L \cdot K^L$ is not large for a small L even when $K(K+1)^n$ is huge).

The fuzzy versions of the two rule evaluation measures (i.e., *confidence* and *support*) described in the previous chapter can be used for candidate rule prescreening. As in Chap. 4 for pattern classification problems, we can find an arbitrary number of candidate rules for modeling problems using a heuristic prescreening criterion. First we generate linguistic rules of length L or less. The confidence and the support are calculated for each of the generated linguistic rules. When L is too large, we are not likely to complete the generation of linguistic rules within the available computation time. On the other hand, when L is too small, we are not likely to generate a large number of good linguistic rules. The value of L should be specified by taking into account various factors such as the available computation time, the number of input variables, the number of given input-output pairs, the number of linguistic terms for input and output variables, etc. As in Chap. 4, the product of the confidence and the support is used as a heuristic prescreening criterion for choosing an arbitrary number of candidate rules from the generated linguistic rules.

9.2.3 Three-Objective Genetic Algorithm for Rule Selection

Let N be the number of candidate rules. As in Chaps. 4 and 6, any subset S of the N candidate rules can be represented by a binary string of length N as

$$S = s_1 s_2 \cdots s_N, \quad (9.14)$$

where $s_q = 1$ and $s_q = 0$ represent the inclusion of the q -th candidate rule R_q in S and the exclusion of R_q from S , respectively.

First a pre-specified number (say N_{pop}) of binary strings of length N are randomly generated as an initial population. The three objectives of each

string S (i.e., rule set S) are evaluated. Copies of non-dominated rule sets are stored as a secondary population separately from the current population. In the application of genetic algorithm-based rule selection to pattern classification problems in Chap. 4, unnecessary rules were removed from each rule set S . While only a single winner rule was used to classify each training pattern in the case of pattern classification problems, all compatible rules are used to calculate the estimated output value for each input vector in modeling problems. If a linguistic rule in S has no compatible input vector, that rule has no effect on the calculation of the estimated output for any input vector in the training data. This means that the removal of such a linguistic rule does not change the value of the first objective $f_1(S)$. Thus we can remove all linguistic rules that satisfy the following condition:

$$\mu_{A_q}(x_p) = 0, \quad p = 1, 2, \dots, m. \quad (9.15)$$

The removal of these linguistic rules improves the second objective $f_2(S)$ and the third objective $f_3(S)$. Usually linguistic rules satisfying (9.15) have already been removed from candidate rules by a prescreening procedure. This is because the prescreening criterion (9.11) always holds if (9.15) holds. Thus we do not use the rule removal procedure based on (9.15) in the genetic algorithm-based rule selection method for modeling problems.

For selecting a pair of parent strings from the current population, the three weights in the fitness function (9.8) are randomly specified as

$$w_i = \text{random}_i / (\text{random}_1 + \text{random}_2 + \text{random}_3), \quad i = 1, 2, 3, \quad (9.16)$$

where random_i is a non-negative random real number. Using binary tournament selection with replacement, a pair of parent strings is selected from the current population based on the fitness function (9.8) with the randomly specified weight values in (9.16). When another pair of parent strings is selected from the current population, the three weights are randomly updated by (9.16). That is, the selection of each pair of parent strings is governed by a different weight vector. From each pair of parent strings, we generate new strings by the uniform crossover and the biased mutation as in the case of pattern classification problems in Chap. 4. By iteratively executing the genetic operations (i.e., selection, crossover, and mutation), we generate $(N_{\text{pop}} - N_{\text{elite}})$ rule sets. The secondary population of non-dominated rule sets is updated using the newly generated rule sets. If a newly generated rule set is not dominated by any other rule sets in the current and secondary populations, its copy is added to the secondary population and all the solutions dominated by the added one are removed from the secondary population. Finally a pre-specified number (say N_{elite}) of non-dominated rule sets are randomly selected from the secondary population and their copies are added to the newly generated $(N_{\text{pop}} - N_{\text{elite}})$ rule sets to form the next population of N_{pop} rule sets. In this manner, the next population is generated from current and secondary populations using the selection, crossover, mutation, and

elitism. The generation update is iterated until a pre-specified stopping condition is satisfied. At each generation, the secondary population is updated to include all the non-dominated rule sets among the examined ones.

9.2.4 Simple Numerical Example

We applied the three-objective genetic algorithm for rule selection to the 441 input–output pairs generated from Fig. 9.2. As in Fig. 9.3, we used the three linguistic terms (i.e., *small*, *medium*, and *large*) for each of the two input and single output variables. The number of possible combinations of antecedent and consequent fuzzy sets is 48. Since the number of possible combinations is very small, we can use all the 48 linguistic rules as candidate rules. In this case, each rule set is represented by a binary string of length 48. The task of the three-objective genetic algorithm for rule selection is to find non-dominated rule sets from the 48 candidate rules. Our computer simulation was performed using the following parameter specifications:

Population size:	$N_{\text{pop}} = 50$,
Number of elite solutions:	$N_{\text{elite}} = 5$,
Crossover probability:	$p_c = 0.8$,
Mutation probability:	$p_m(0 \rightarrow 1) = 1/48$, $p_m(1 \rightarrow 0) = 0.1$,
Stopping condition:	1000 populations.

After 1000 iterations of the population update using the genetic operations, five rule sets were obtained (i.e., these rule sets were included in the secondary population after the 1000th generation). The obtained rule sets are shown in Table 9.1.

Table 9.1. Non-dominated rule sets for the nonlinear function in Fig. 9.2

Rule set	Total squared error	Number of rules	Average rule length
S_1	441	0	-
S_2	27.6	1	0.0
S_3	7.4	2	0.5
S_4	3.7	3	1.0
S_5	0.0	4	1.25

The simplest rule set S_2 (excluding an empty set S_1) in Table 9.1 includes only a single linguistic rule of length 0. The linguistic rule is

$$R_1: y \text{ is } \textit{medium}. \quad (9.17)$$

This is a very rough description of the nonlinear function in Fig. 9.2. The second simplest rule set S_3 includes the following linguistic rule in addition to R_1 in (9.17).

$$R_2: \text{If } x_1 \text{ is } \textit{large} \text{ then } y \text{ is } \textit{small}. \quad (9.18)$$

In Fig. 9.4, we show the nonlinear function depicted from the rule set $S_3 = \{R_1, R_2\}$ using the non-standard fuzzy reasoning method. From the comparison between Fig. 9.4 and Fig. 9.2, we can see that the rule set S_3 is a rough approximation of the nonlinear function in Fig. 9.2. The accuracy of approximation can be improved by increasing the number of linguistic rules. For example, the rule set S_5 in Table 9.1 includes four linguistic rules, which are shown in Fig. 9.5. The corresponding estimated nonlinear function is shown in Fig. 9.6. From the comparison between Fig. 9.6 and Fig. 9.2, we can see that the rule set S_5 with the four linguistic rules in Fig. 9.5 approximates the nonlinear function in Fig. 9.2 very well (actually the total squared error is zero as shown in Table 9.1). As we can see from Table 9.1, there exists a tradeoff between the accuracy and the complexity of linguistic rule-based systems.

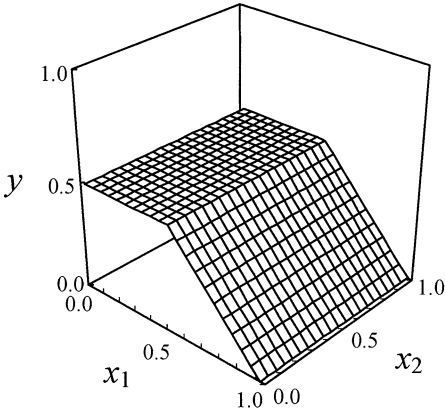


Fig. 9.4. Estimated nonlinear function from the obtained rule set S_3

In the above computer simulation, we used all the 48 linguistic rules as candidate rules for rule selection. As we have already explained, eight linguistic rules have no compatible input-output pairs. Thus these linguistic rules can be removed from candidate rules. We performed the same computer simulation using the remaining 40 linguistic rules as candidate rules. Exactly the same rule sets as Table 9.1 were obtained from this computer simulation.

9.3 Fuzzy Genetics-Based Machine Learning

In the application of linguistic rule-based systems to pattern classification problems, it is easy to implement a Michigan-style genetics-based machine learning (GBML) algorithm as shown in Chap. 5. This is because a single winner rule is responsible for the classification of each training pattern. A

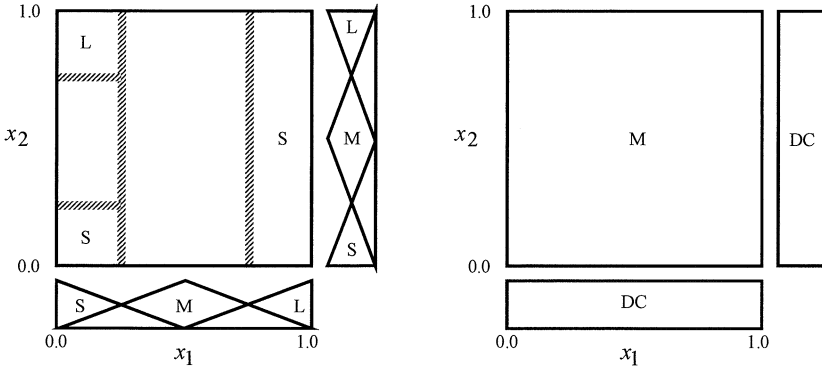


Fig. 9.5. Obtained rule set S_5 with four linguistic rules in Table 9.1 for the non-linear function in Fig. 9.2

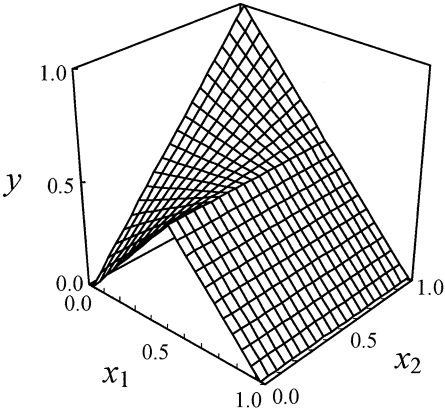


Fig. 9.6. Estimated nonlinear function from the obtained rule set S_5 in Fig. 9.5

reward or penalty will be given to the single winner rule depending on the classification result (i.e., correct classification or misclassification). On the other hand, the definition of a fitness function for each linguistic rule is not easy in modeling problems because multiple linguistic rules are involved in the calculation of the estimated output value for each input vector. Moreover, the evaluation of the estimated output value is not easy, while the evaluation of the classification result is straightforward (i.e., correct classification or misclassification). In this section, we implement a Pittsburgh-style GBML algorithm for the design of linguistic rule-based systems for modeling problems. See [13] for recent developments of Michigan-style fuzzy GBML algorithms (i.e., fuzzy classifier systems).

9.3.1 Coding of Rule Sets

For simplicity of explanation, let us assume that we have the three linguistic terms (i.e., *small*, *medium*, and *large*) for each of the n input and single output variables as in Fig. 9.3. In the same manner as in Chap. 5, we use the following four symbols to denote the four antecedent fuzzy sets:

- 1: *small*,
- 2: *medium*,
- 3: *large*,
- #: *don't care*.

It should be noted that *don't care* is not used as a consequent fuzzy set (i.e., it is used only in the antecedent part). The total number of combinations of antecedent and consequent fuzzy sets is $(3 + 1)^n \cdot 3$. Each combination corresponds to a single linguistic rule. In Pittsburgh-style fuzzy GBML algorithms, each linguistic rule R_q in (9.1) is coded as a substring of length $(n + 1)$ using its antecedent and consequent fuzzy sets as $A_{q1}A_{q2} \dots A_{qn}B_q$. For example, a substring “1##232” denotes the following linguistic rule for a modeling problem with five input variables:

Rule R_1 : If x_1 is *small* and x_4 is *medium* and x_5 is *large*
then y is *medium*. (9.19)

Each rule set S including N_{rule} linguistic rules is represented by a concatenated string of length $(n + 1) \cdot N_{\text{rule}}$ and handled as an individual in our Pittsburgh-style fuzzy GBML algorithm. For example, a rule set of the following four linguistic rules for a modeling problem with five input variables is represented by a concatenated string “1##232 22####1 #####12 3#####33” of length 24:

Rule R_1 : If x_1 is *small* and x_4 is *medium* and x_5 is *large*
then y is *medium*, (9.20)

Rule R_2 : If x_1 is *medium* and x_2 is *medium* then y is *small*, (9.21)

Rule R_3 : If x_5 is *small* then y is *medium*, (9.22)

Rule R_4 : If x_1 is *large* and x_5 is *large* then y is *large*. (9.23)

9.3.2 Three-Objective Fuzzy GBML Algorithm

In our three-objective fuzzy GBML algorithm for approximately realizing an n -input and single-output function, each rule set including N_{rule} linguistic rules is represented by a string of length $(n + 1) \cdot N_{\text{rule}}$. First we randomly generate a pre-specified number (say N_{pop}) of strings of length $(n + 1) \cdot N_{\text{rule}}$. Then the three objectives of each string S (i.e., rule set S) are evaluated. Copies of non-dominated rule sets are stored as a secondary population separately from the current population.

The selection operation of parent strings is the same as the previous section. That is, the three weights in the fitness function (9.8) are randomly specified for selecting a pair of parent strings as shown in (9.16). We use binary tournament selection with replacement. When another pair of parent strings is selected from the current population, the three weights are randomly updated by (9.16).

From each pair of parent strings, we generate new strings using crossover and mutation. We use the one-point crossover operation with different crossover points as in Chap. 5. This crossover operation is illustrated in Fig. 9.7 (also see Fig. 5.15).

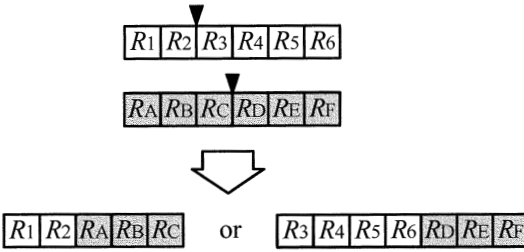


Fig. 9.7. One-point crossover operation with different crossover points

Two mutation operations are applied to each of the generated new strings by the crossover operation. One mutation operation randomly replaces an antecedent or consequent fuzzy set with another one. Note that *don't care* is not used as a consequent fuzzy set, though it is used as an antecedent fuzzy set. This mutation operation is applied to each antecedent or consequent fuzzy set with a pre-specified mutation probability. The other mutation operation is to randomly remove linguistic rules from each rule set. This mutation operation is applied to each linguistic rule with a pre-specified mutation probability. These mutation operations can also be used in the fuzzy GBML algorithms in Chaps. 5 and 6 for classification problems.

By iteratively executing the genetic operations (i.e., selection, crossover, and mutation), we generate $(N_{\text{pop}} - N_{\text{elite}})$ rule sets. The secondary population of non-dominated rule sets is updated using the newly generated rule sets. Finally a pre-specified number (say N_{elite}) of non-dominated rule sets are randomly selected from the secondary population and their copies are added to the newly generated $(N_{\text{pop}} - N_{\text{elite}})$ rule sets to form the next population of N_{pop} rule sets. In this manner, the next population is generated from the current and secondary populations using selection, crossover, mutation, and elitism. The generation update is iterated until a pre-specified stopping condition is satisfied. At each generation, the secondary population is updated to include all the non-dominated rule sets among the examined ones.

9.3.3 Simple Numerical Example

We applied the three-objective fuzzy GBML algorithm to the 441 input-output pairs generated from Fig. 9.2 to design linguistic rule-based systems. As in the previous computer simulation in Sect. 9.2.4, we used the three linguistic terms (i.e., *small*, *medium*, and *large*) for each of the two input and single output variables. Our computer simulation was performed using the following parameter specifications:

Population size: 50,
 Number of linguistic rules in each initial rule set: 10,
 Number of elite solutions: 5,
 Crossover probability: 0.8,
 Mutation probability: 0.05 for replacement of each fuzzy set,
 0.05 for removal of each linguistic rule,
 Stopping condition: 1000 populations.

After 1000 iterations of the population update using the genetic operations, five rule sets were obtained (i.e., these rule sets were included in the secondary population after the 1000th generation). The obtained rule sets are exactly the same as those in Table 9.1.

9.3.4 Some Heuristic Procedures

As we have already mentioned, the total number of possible linguistic rules is $K \cdot (K + 1)^n$ when K linguistic terms are given for each of the n input and single output variables. Each substring of length $(n + 1)$ corresponds to one of the $K \cdot (K + 1)^n$ linguistic rules. Since each rule set S (i.e., each string S) is a subset of the $K \cdot (K + 1)^n$ linguistic rules, the size of the search space is 2^N where $N = K \cdot (K + 1)^n$. This means that the size of the search space rapidly increases as the number of input variables increases. Thus the three-objective fuzzy GBML algorithm in this section does not always work well on high-dimensional problems while it worked well on the simple numerical example in the previous computer simulation.

We explain two heuristic procedures [85] for improving the search ability of the three-objective fuzzy GBML algorithm to find good rule sets for high-dimensional modeling problems. Both heuristic procedures are applied to rule sets generated by the genetic operations in each generation.

One is a heuristic replacement procedure of the consequent linguistic term of each linguistic rule. Since the genetic operations do not take into account the given input-output pairs, each linguistic rule does not always have an appropriate consequent linguistic term. The heuristic replacement procedure probabilistically replaces the consequent linguistic term of each linguistic rule with a more appropriate one using the information about the given input-output pairs. The replacement probability for each linguistic term from the current one is defined using the confidence as

$$p(B_k) = \frac{c(\mathbf{A}_q \Rightarrow B_k)}{\sum_{k=1}^K c(\mathbf{A}_q \Rightarrow B_k)}, \quad (9.24)$$

where K is the number of linguistic terms for the output variable. This procedure is applied to each linguistic rule with a pre-specified probability. In the computer simulations in the next section, the application probability is specified as 0.5.

Let us consider a linguistic rule of the form “If x_1 is *small* and x_2 is *small* then y is B_q ” for the simple numerical example in Fig. 9.2 with the 441 input–output pairs. The confidence of each linguistic rule with the antecedent part (*small, small*) is calculated as

$$c((small, small) \Rightarrow small) = 0.49, \quad (9.25)$$

$$c((small, small) \Rightarrow medium) = 0.51, \quad (9.26)$$

$$c((small, small) \Rightarrow large) = 0.00. \quad (9.27)$$

Thus the consequent part of a linguistic rule of the form “If x_1 is *small* and x_2 is *small* then y is B_q ” is replaced with *small* or *medium* when the heuristic replacement procedure is applied to the linguistic rule. The replacement probabilities of *small* and *medium* are 0.49 and 0.51, respectively. That is, *small* and *medium* are chosen as the consequent fuzzy set with the probabilities of 0.49 and 0.5, respectively.

The other heuristic procedure is to generate a linguistic rule from an input–output pair with the maximum error. A similar idea was explained for pattern classification problems in Chap. 5. In this procedure, first the squared error for each input–output pair is calculated using each rule set in the current population after the replacement procedure. Next an input–output pair with the maximum error is identified for each rule set. Let (\mathbf{x}_p, y_p) be the input–output pair with the maximum error for the rule set S . Then a linguistic rule is generated from this input–output pair. Its antecedent and consequent parts are determined by the most compatible linguistic terms with the input and output values (\mathbf{x}_p, y_p) . For example, if $\mathbf{x}_p = (0.12, 0.48, 0.97)$ and $y_p = 0.57$, then the following linguistic rule is generated:

$$\begin{aligned} &\text{If } x_1 \text{ is } small \text{ and } x_2 \text{ is } medium \text{ and } x_3 \text{ is } large \\ &\quad \text{then } y \text{ is } medium. \end{aligned} \quad (9.28)$$

Each antecedent linguistic term is replaced with *don't care* using a pre-specified probability. This probability is specified as 0.5 in the computer simulations in the next section. The generated linguistic rule is added to the rule set S . This procedure is applied to each rule set with a pre-specified probability (0.1 in the computer simulations in the next section) after the above-mentioned replacement procedure.

9.4 Comparison of Two Schemes

We compared the following four algorithms with each other through computer simulations on numerical examples:

- (1) The genetic algorithm-based rule selection method with no prescreening procedure.
- (2) The genetic algorithm-based rule selection method with a heuristic prescreening procedure.
- (3) The fuzzy GBML algorithm with no heuristic procedure.
- (4) The fuzzy GBML algorithm with the two heuristic procedures in Sect. 9.3.4.

In the second algorithm, 100 candidate rules were selected using the product of the confidence and the support as a heuristic prescreening criterion. When the total number of candidate rules was not more than 100, all candidate rules were used in rule selection.

Since the evaluation of simulation results by multi-objective optimization methods is not easy, we used these four algorithms as single-objective optimization methods by specifying the three weights as $w_1 = 100$, $w_2 = 1$, and $w_3 = 1$. That is, we used the following fitness function in the four algorithms:

$$fitness(S) = -100f_1(S) - f_2(S) - f_3(S). \quad (9.29)$$

As training data, we generated 441 input–output pairs (x_{p1}, x_{p2}, y_p) , $p = 1, 2, \dots, 441$, from the nonlinear function in Fig. 9.8 using the uniformly divided 21×21 grid of the input space $[0, 1] \times [0, 1]$. That is, $x_{p1} = 0.00, 0.05, \dots, 1.00$, $x_{p2} = 0.00, 0.05, \dots, 1.00$, and the value of y_p was calculated from Fig. 9.8. For each of the two input and single output variables, we used five linguistic terms (i.e., *small*, *medium small*, *medium*, *medium large*, and *large*).

We also generated a test problem with three input variables by adding an additional input variable x_3 to the test problem with the two input variables. The value of x_3 in each of the 441 input–output pairs $(x_{p1}, x_{p2}, x_{p3}, y_p)$, $p = 1, 2, \dots, 441$, was randomly specified as a real number in the unit interval $[0, 1]$. In the same manner, we also generated test problems with four and five input variables.

The length of bit strings in the genetic algorithm-based rule selection (i.e., the first algorithm) becomes extremely long as the number of input variables increases. This is because the number of candidate rules exponentially increases as the number of input variables increases. It is impractical to directly apply the first algorithm to the problems with three and more input variables. Thus, we modify the initialization process of the first algorithm. We pre-specified the probability of $s_i = 1$ for $i = 1, \dots, N$ in (9.14) as 0.1 in the case of the three and the four input variables and 0.001 in the case of the five input variables. This modification reduces the CPU time of the

algorithm since the number of selected rules in the initial population does not become large.

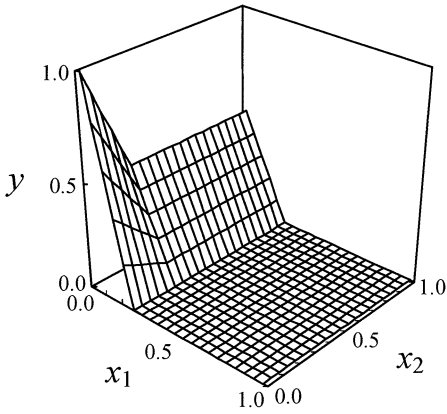


Fig. 9.8. A test problem

We applied each of the four algorithms to each of the four test problems ten times. Average simulation results are summarized in Tables 9.2 – 9.5. In these tables, each algorithm was implemented in the C language and executed on a PC with a Pentium IV 2.80 GHz processor. From the comparison between Table 9.2 and Table 9.3, we can see that the prescreening procedure of candidate rules significantly decreased the CPU time of the genetic algorithm-based rule selection method. As we have already demonstrated in Chapt. 4 for pattern classification problems, candidate rule prescreening is necessary in rule selection for handling high-dimensional problems. However, the performance of the genetic algorithm-based rule selection method with the prescreening procedure is significantly worse than that with no prescreening procedure in some cases. This is because we cannot always successfully select important linguistic rules as candidate rules for the genetic algorithm-based rule selection method by the prescreening procedure. For example, in the case of the four input variables, we did not select the following linguistic rule as a candidate rule:

$$\text{If } x_1 \text{ is } \textit{small} \text{ and } x_2 \text{ is } \textit{small} \text{ then } y \text{ is } \textit{large}. \quad (9.30)$$

The product of the support and the confidence of this linguistic rule is 0.001. This value is the 974th largest in the entire set of the generated linguistic rules. It should be noted that 100 candidate rules were selected in the computer simulations for Table 9.3.

From the comparison between Table 9.4 and Table 9.5, we can see that the search ability of the fuzzy GBML algorithm was improved by the two heuristic procedures. We can also see from Tables 9.2 – 9.5 that better results were obtained by the fuzzy GBML algorithm in Table 9.4 and Table 9.5 than the rule selection method in Table 9.2 and Table 9.3.

Table 9.2. Average simulation results over ten trials for each test problem using the genetic algorithm-based rule selection method with no candidate rule prescreening procedure

Number of input variables	Total squared error	Number of rules	Average rule length	CPU time (s)
2	0.019	3.4	1.03	250.2
3	0.114	4.5	1.18	1006.7
4	0.197	4.1	1.10	12103.1
5	1.074	7.8	2.08	35316.8

Table 9.3. Average simulation results over ten trials for each test problem using the genetic algorithm-based rule selection method with the prescreening procedure

Number of input variables	Total squared error	Number of rules	Average rule length	CPU time (s)
2	0.0	3.0	1.00	163.6
3	0.833	3.0	1.00	155.2
4	0.870	2.0	0.50	146.8
5	8.596	1.0	0.00	102.5

Table 9.4. Average simulation results over ten trials for each test problem using the fuzzy GBML algorithm with no heuristic procedure

Number of input variables	Total squared error	Number of rules	Average rule length	CPU time (s)
2	0.0	3.0	1.00	72.5
3	0.072	3.7	1.11	269.5
4	0.051	3.1	0.97	505.6
5	0.190	3.7	1.16	1243.6

Table 9.5. Average simulation results over ten trials for each test problem using the fuzzy GBML algorithm with the two heuristic procedures

Number of input variables	Total squared error	Number of rules	Average rule length	CPU time (s)
2	0.0	3.0	1.00	597.3
3	0.0	3.0	1.00	4829.2
4	0.0	3.2	1.13	8330.3
5	0.026	3.2	1.13	11010.0

Classification and Modeling with Linguistic Information
Granules

Advanced Approaches to Linguistic Data Mining

Ishibuchi, H.; Nakashima, T.; Nii, M.

2005, XII, 308 p., Hardcover

ISBN: 978-3-540-20767-2